A hybrid Constraint Programming/Mixed Integer Programming framework for the preventive signaling maintenance crew scheduling problem

Pour, Shahrzad M.; Drake, John H.; Ejlertsen, Lena Secher; Rasmussen, Kourosh Marjani; Burke, Edmund K.

Published in:
European Journal of Operational Research

Link to article, DOI:
10.1016/j.ejor.2017.08.033

Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

European Journal of Operational Research 269 (2018) 341–352

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

A hybrid Constraint Programming/Mixed Integer Programming framework for the preventive signaling maintenance crew scheduling problem

Shahrzad M. Pour[a,*], John H. Drake[b], Lena Secher Ejlertsen[c], Kourosh Marjani Rasmussen[a], Edmund K. Burke[b]

[a] DTU Management Engineering, Technical University of Denmark, Produktionstorvet, 2800 Kgs. Lyngby, Denmark
[b] Operational Research Group, Queen Mary University of London, Mile End Road, London E1 4NS, UK
[c] Banedanmark, Amerika Plads 15 DK-2100 Copenhagen, Denmark

ARTICLE INFO

Article history:
Received 28 October 2016
Accepted 16 August 2017
Available online 31 August 2017

Keywords:
Transportation
Scheduling
Constraint programming
Mixed Integer Programming
Hybrid approaches

ABSTRACT

A railway signaling system is a complex and interdependent system which should ensure the safe operation of trains. We introduce and address a mixed integer optimisation model for the preventive signal maintenance crew scheduling problem in the Danish railway system. The problem contains many practical constraints, such as temporal dependencies between crew schedules, the splitting of tasks across multiple days, crew competency requirements and several other managerial constraints. We propose a novel hybrid framework using Constraint Programming to generate initial feasible solutions to feed as 'warm start' solutions to a Mixed Integer Programming solver for further improvement. We apply this hybrid framework to a section of the Danish rail network and benchmark our results against both direct application of a Mixed Integer Programming solver and modelling the problem as a Constraint Optimisation Problem. Whereas the current practice of using a general purpose Mixed Integer Programming solver is only able to solve instances over a two-week planning horizon, the hybrid framework generates good results for problem instances over an eight-week period. In addition, the use of a Mixed Integer Programming solver to improve the initial solutions generated by Constraint Programming is shown to be significantly superior to addressing the problem as a Constraint Optimisation Problem.

1. Introduction

A railway signaling system is an essential component of a railway network. It is, responsible for ensuring safe and efficient train operations. The existing signaling technology within the Danish railway network is based on the Automatic Train Protection (ATP) signaling system (Banedanmark. & Trafikministeriet., 2009). To ensure that signaling equipment is both cost efficient and safe throughout its service life, effective maintenance planning is crucial. Generally, railway maintenance planning and scheduling problems are considered as either strategic, tactical or operational level problems (Lidén, 2015). Using this terminology, the problem that we consider here is considered to be a tactical problem, where the aim is to assign and schedule a set of maintenance tasks to maintenance crew members over a given planning horizon. Additionally, there are several aspects which could differ from one railway network to another, such as the competency level required for fulfilling each task, coordination with train traffic, transportation related costs, and several hard and soft managerial constraints.

A number of papers exist in the literature that address maintenance crew scheduling, with a variety of formulations and solution techniques proposed. Cheung, Chow, Hui, and Yong (1999) presented a Constraint Programming (CP) model for scheduling maintenance tasks within the Hong Kong Mass Transit system. The results showed that the proposed CP method was 10 times more efficient than the existing manual method used in practice. Gorman and Kanet (2010) developed a time-space network model and a job scheduling model to schedule maintenance tasks, showing results for a small test instance. The first model was solved as a Mixed Integer Programming (MIP) problem, with the second model solved using a hybrid Constraint Programming and Genetic Algorithm approach. Nemani, Bog, Ahuja, 2010 proposed four different models for the curfew planning problem, which adds mutual exclusion and

* Corresponding author.
E-mail addresses: shmp@dtu.dk, shahrzad.mpour@gmail.com (S. M. Pour), j.drake@qmul.ac.uk (J.H. Drake), lsej@bane.dk (L.S. Ejlertsen), kmra@dtu.dk (K.M. Rasmussen), e.burke@qmul.ac.uk (E.K. Burke).

http://dx.doi.org/10.1016/j.ejor.2017.08.033

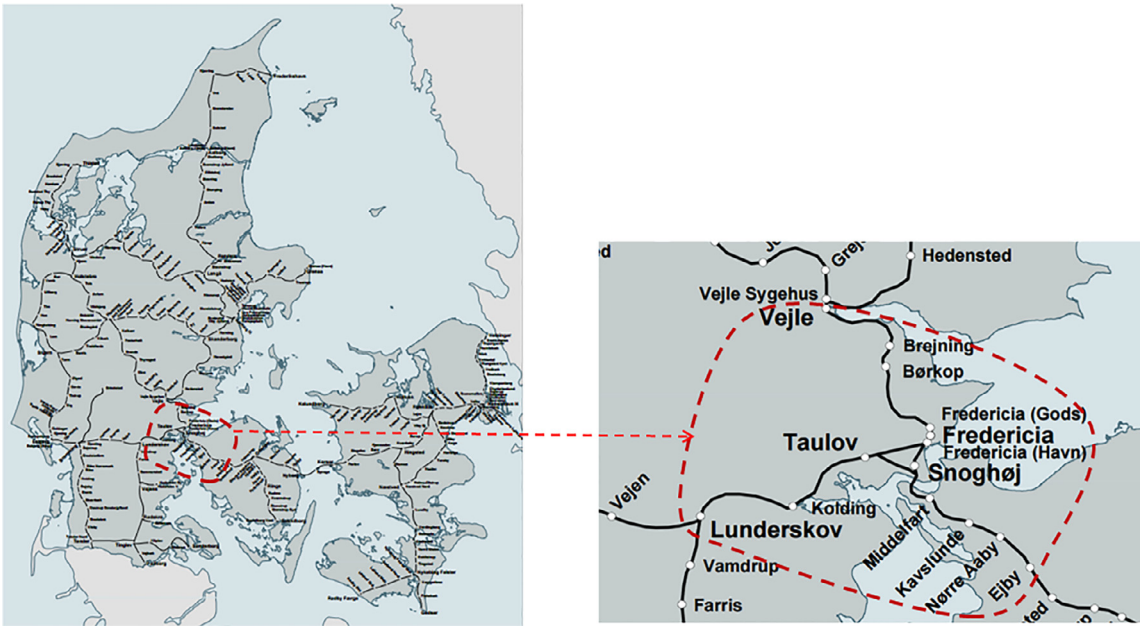

Fig. 1. Pilot area of the signaling maintenance problem in Denmark.

time window constraints to the core problem of scheduling tasks. Each model was solved with a commercial MIP solver, using real-world instances from a large rail company. Bog, Nemani, and Ahuja (2011) also solved the curfew planning problem. Their method iteratively solved sub-problems using a MIP solver, gradually increasing the size of the sub-problem until the entire instance was included. This method was applied to the instances used by Nemani, Bog, Ahuja, 2010, outperforming three of the four approaches from their paper. Peng et al. (2011) presented a cluster-first, route-second approach to minimise the travel cost of maintenance teams. An initial phase provides an assignment of tasks to maintenance teams before a local search phase attempts to improve the solution found. Their results showed a significant improvement over manual planning. A two-phase approach was used by Borraz-Sánchez and Klabjan (2012), first applying dynamic programming to generate an initial schedule, before a second phase of improvement with a *ruin and recreate* heuristic (Schrimpf, Schneider, Stamm-Wilbrandt, & Dueck, 2000) using an ILP model to reinsert tasks optimally. Their method was able to solve an annual scheduling problem with 1000 tasks within 2.5 hours. Peng and Ouyang (2014) described a method which combines multiple maintenance tasks into longer projects as a pre-processing stage before allocating the tasks to maintenance crew. The proposed model is also solved by a method performing an initial constructive phase before a second phase of local improvement, and was adopted in practice by the company providing the case study. Khalouli, Benmansour, and Hanafi (2016) presented an ant colony method to address a set of randomly generated instances of the preventive maintenance scheduling problem. The proposed method was able to generate optimal solutions to some instances in significantly less time than that required by a commercial MIP solver. Wen, Li, and Salling (2016) formulated the problem of determining when to perform 'tamping', a track maintenance operation, on different sections of a railway network as a MIP model. Baldi, Heinicke, Simroth, and Tadei (2016) consider a stochastic variant of the tactical railway maintenance problem, where the exact maintenance tasks required to be performed are not known in advance, and scheduling takes place over a long-term rolling planning horizon.

As the infrastructure owner of most of the rail network in Denmark, Banedanmark is in charge of the maintenance and traffic control of the Danish railway track and signaling system. The Danish rail network comprises four maintenance areas: Maintenance Machines, Maintenance Nationwide, Maintenance East and Maintenance West. The East and West divisions are further divided into Track Maintenance, Signaling Maintenance and Current Maintenance. The pilot maintenance region that we consider in this paper is part of the signaling section of the West region. It is situated between Ejby, Lunderskov and Vejle as shown in Fig. 1. The current practice is to produce plans over a two-week planning horizon using a commercial MIP solver.

The main contribution of this paper is the formulation of the preventive signaling maintenance crew scheduling problem for the existing signaling system in Denmark as a mixed integer optimisation model. The crew start their tasks from a depot location. Three characteristics of the problem add to the complexity of the model. Firstly, the plan includes temporal dependencies between different crew members. That is because some of the tasks require more than one crew member, due to crew competency requirements or safety rules. Secondly, to handle the considerations that must be made for traffic, multiple crew members can fulfil a task together to minimise the possession time of the track. Accordingly, there is a range in terms of the number of crew members required to fulfil a given task per day. Finally, the majority of tasks take much longer than a single day, even with multiple crew members working on them, requiring a plan to be split over multiple days.

For the real-world problem, monthly plans are expected for operational reasons and currently optimal solutions cannot be found for practical sized problem instances. Here, we introduce a hybrid framework, using CP to generate initial feasible solutions to feed to a MIP solver for further improvement.

The remainder of the paper is structured as follows: in Section 2, we describe the MIP formulation of the problem and explain the real-life constraints within the model. Section 3 explains our solution approach. In Section 4, the details of the real-world instances used are given and results for the proposed hybrid framework are presented. Finally we provide some conclusions in Section 5.

2. Mathematical model

The model formulation is provided by Banedanmark and is based on the practical maintenance crew scheduling problem

encountered by the Banedanmark planning team. The problem consists of a number of *technical places* where maintenance tasks are required to be carried out. A technical place is either a station or the maintenance area between a station and the next station. The crew start their tasks from a depot location and return to the depot at the end of every day. The model covers travelling distance to and from the depot, transportation costs between technical places during the working day and the duration of maintenance tasks, with the hard constraint that the plan does not exceed the maximum shift length each day. The model also considers that crew members should have the correct competence level for a particular task and it defines the minimum and maximum number of crew members that can work simultaneously on each task. For longer tasks that are completed over more than one shift, it is desirable to allocate the same crew members to continue the task the next day. The model in its entirety is explained in the following subsections. Within the model, M represents an arbitrarily large number to help bound some of the constraints.

2.1. Indexes

n	crew $n \in [N]$
i	task $i \in [I]$
j	date $j \in [J]$
k	competencies $k \in [K]$
$p, (q)$	technical place $p \in [P]$

2.2. Parameters

a	number of hours per shift
f	total competence level needed
c_i	time required to complete task i
$d1_i$	minimum number of crew for task i
$d2_i$	maximum number of crew for task i
e_{nj}	whether crew member n is available on planning date j
bo_{ik}	whether task i demands competence k
bm_{nk}	whether crew n has at least competence level 3 for competence k
$bm2_{nk}$	1 if crew n has less than competence level 3 for competence k
$bm3_{nk}$	competence level for crew n for competence k
tp_{ip}	if task i is physically located at technical place p
tr_{pq}	transport time from technical place p to technical place q
tm_p	transport time from depot to technical place p
g_i	1 if the task must be done inside the planning horizon, 0 if it can be left out

2.3. Variables

x_{nij}	fraction of task i that crew n completes on date j.
$x3_{ij}$	fraction of task i that is completed on date j.
$x2_{ij}$	1 if some of task$i$is completed on date$j$ 0 else
$x4_i$	1 if task$i$is fully completed within the planning horizon 0 else
$x5_{nij}$	1 if crew member$n$is working on task$i$on date$j$but not on date$j+1$ 0 else
$x6_{ij}$	1 if part of task$i$is completed on date$j$but not on date$j+1$ 0 else
y_{nj}	1 if crew member$n$will work on date$j$ 0 else
z_{nij}	1 if crew member$n$works on task$i$on date$j$ 0 else
$z1_{ni}$	1 if crew$n$works on task$i$ 0 else
w_{npj}	1 if crew$n$works on technical place$p$on date$j$ 0 else
v_{npqj}	1 if crew$n$needs transport between technical place$p$and technical place$q$on date$j$ 0 else
$w1_{npj}$	if crew n needs transport to technical place p from another technical place on date j
$w2_{npj}$	if crew n needs transport from a technical place p to another technical place on date j

2.4. Objective function

The objective function is primarily composed of three parts. Firstly, it aims to minimise the number of working days used to complete the plan. Secondly, it should ensure that as many tasks as possible are completed inside the planning horizon. Finally, the model tries to minimise the penalty for assigning crew members to a particular task on non-consecutive days. In order to normalise this multi-objective function we have scaled each term, dividing it by the maximum possible value for that specific term. The weighted sum method is applied to give relative coefficients/weights to each term of the objective function. The sum of the weights are one and are provided by the planning manager from Banedanmark to reflect the importance of each to the company. Priority is given in the following order: fulfilling a greater number of tasks in the planning time horizon, minimising the total number of working days and finally, generating a high quality plan from a managerial point of view.

$$\min O = \sum_n \sum_j y_{nj} \cdot a + \sum_{nij} z_{nij} + \sum_{ni} z1_{ni} + \sum_{nij} x5_{nij} + \sum_{ij} x6_{ij} + \sum_n \sum_{j=5} y_{nj} - \sum_n \sum_{j=1} y_{nj} + \sum_i (1 - x4_i) \cdot c_i \quad (1)$$

2.5. Constraints

2.5.1. Constraints in relation to the tasks

All tasks should either be completed entirely or not completed at all within the planning horizon:

$$\sum_n \sum_j x_{nij} = x4_i \quad \forall \quad i \quad (2)$$

The total number of hours for each shift should not be exceeded. The first term is the duration of tasks, the second term is the transportation time to and from the depot, and third term is the transportation time between technical places during the shift:

$$\sum_i x_{nij} \cdot c_i + \sum_p (w_{npj} \cdot 2 - w1_{npj} - w2_{npj}) \cdot tm_p + \sum_p \sum_q v_{npqj} \cdot tr_{pq} \le a \quad \forall \quad j, n \quad (3)$$

The sum of the fractions of tasks allocated to crew members cannot exceed the total required to complete the task:

$$x2_{ij} \ge \sum_n x_{nij} \quad \forall \quad i, j \quad (4)$$

$x3$ is defined as the sum of the fractions of a task allocated to all crew members for a particular task on a given day:

$$x3_{ij} = \sum_n x_{nij} \quad \forall \quad i, j \quad (5)$$

Some tasks are considered to be critical and must be completed inside the planning horizon, meaning that they are high priority. The more tasks that are fulfilled, the better the plan is considered

to be. Accordingly, a task i must be completed within the planning horizon if parameter g_i is set to 1:

$$x4_i \geq g_i \quad \forall \quad i \tag{6}$$

If a task is completed within the planning horizon, the fraction of a task that is completed on a given day should not exceed $x4$:

$$x4_i \geq x_{nij} \quad \forall \quad n, i, j \tag{7}$$

A crew member cannot be allocated a task on a day that they are not due to work:

$$y_{nj} \geq z_{nij} \quad \forall \quad n, i, j \tag{8}$$

If a crew member is allocated a fraction of a task on a particular date, Eq. (9) ensures that the variable indicating that a crew member is working on this task on this date is set to 1. Eq. (10) ensures that this variable cannot be set to 1 if the crew member is not allocated a fraction of this task on a particular date.

$$z_{nij} \geq x_{nij} \quad \forall \quad n, i, j \tag{9}$$

$$z_{nij} \leq x_{nij} \cdot M \quad \forall \quad n, i, j \tag{10}$$

If a crew member is allocated a fraction of a task to complete on a particular date, the variable indicating if a crew member works on this task at all should always at least as large as this value:

$$z1_{ni} \geq z_{nij} \quad \forall \quad n, i, j \tag{11}$$

2.5.2. Managerial constraints

From a managerial point of view, if a given task takes more than a day to complete, then the following soft constraints will be desired:

- If some crew members work on a task on date j but do not continue the following day, then the remaining parts of the task should preferably be undertaken by the same remaining crew members who started working on the task:

$$x5_{nij} \geq z_{nij} - z_{nij+1} \quad \forall \quad n, i, j \tag{12}$$

- If task i is started but not completed on date j and is not continued the following day, resulting in the task being fulfilled on non-consecutive days, then a penalty will be given to the plan:

$$x6_{ij} \geq x2_{ij} - x2_{ij+1} \quad \forall \quad i, j \tag{13}$$

2.5.3. Constraints in relation to the crew

According to Banedanmark, the suggested plan should allow for assigning multiple crew members to one task in order to shorten the total time that it takes to complete. On the other hand, having too many employees working on each task weakens the sense of responsibility and therefore the quality of the job done by crew members. As a result, Banedanmark provides a maximum possible number of crew members which can be assigned to each task. In addition, due to safety regulations there are some tasks that require at least two crew members to work on them simultaneously. Therefore, there is a minimum and maximum number of crew members that can work simultaneously on a task on a given date.

The minimum number of crew members that should work (simultaneously) on a task per date is represented by:

$$\sum_n z_{nij} \geq d1_i \cdot x2_{ij} \quad \forall \quad i, j \tag{14}$$

Similarly, the maximum number of crew members that should work (simultaneously) on a task per date is represented by:

$$\sum_n z_{nij} \leq d2_i \cdot x2_{ij} \quad \forall \quad i, j \tag{15}$$

Each crew member cannot perform more than the fraction of a task that can be completed by the minimum number of crew members required. This ensures that at least the minimum number of crew members required work on each task simultaneously:

$$x_{nij} \leq \frac{x3_{ij}}{d1_i} \quad \forall \quad n, i, j \tag{16}$$

As crew members will not available for all dates due to working shift patterns, vacation, training etc., crew members cannot be assigned to work on a task on a date that they are not due to work:

$$z_{nij} \leq e_{nj} \quad \forall \quad n, i, j \tag{17}$$

2.5.4. Constraints in relation to competencies

The model also considers that crew members must have the right competence level to complete different tasks. We believe that satisfying the competencies required for each task is the most challenging part of the model, since the number of crew working on each task is not predetermined in advance and can vary within a possible range. This is further complicated by the fact that tasks can be split over multiple days. As a result, the number of crew members needed to satisfy the crew competency requirements can change based on the number of crew working on a task per day.

In order to satisfy the crew competency requirements for each task, there are three possible acceptable scenarios defined by the planners. Fig. 2 shows the scenarios which lead to the crew competency requirements being met. We suppose that there is a task called *task*1 which demands crew with competency level 3 of *A* and there are two crew members *crew*1 and *crew*2 with competencies level 3 of *A* and less than level 3 of *A*, respectively.

- When the minimum number of crew required for fulfilling *task*1 is one person, there are two possible states:
  - One crew member is assigned to the task. *Crew*1 is assigned to *Task*1 and 100% of the task is undertaken by the same person (a).
  - More than one crew member is assigned to the task. *Crew*1 and *Crew*2 are assigned to *Task*1. Since *Crew*2 does not have the required competency level 3 for undertaking *Task*1, they can only work on the task simultaneously with *Crew*1. *Crew*1 can fulfil the remaining part of the task on his own due to his level of competency (b). What is crucial is satisfying the level of competency until a task is finished. The process of accomplishing the task will be shortened by having more than one crew member involved.
- If *Task*1 needs crew competency *A* and the minimum number of crew required is two persons, it necessitates that both crew members attend simultaneously (c).

To summarise, at least one of the crew members should have the right competence level for a task and the minimum and maximum number of crew members that can be allocated to a task should be respected. For the particular scheduling problem at hand, each crew member has a competence level ranging from 0 to 4. A crew member is considered as an expert if they have at least level 3 for a particular competency and at least one expert crew member should be present at all times when working on a specific task. The total competence level f of crew members working simultaneously on a task should be at least 4.

On this basis, the related constraints are defined as follows. The combined competence level of all crew members should be sufficient for each task:

$$\sum_n z_{nij} \cdot bm3_{nk} \geq x2_{ij} \cdot bo_{ik} \cdot f \quad \forall \quad i, j, k \tag{18}$$

At least one crew member should have competence level 3 for the equipment type of task i:

$$\sum_n z_{nij} \cdot bm_{nk} \geq x2_{ij} \cdot bo_{ik} \quad \forall \quad i, j, k \tag{19}$$

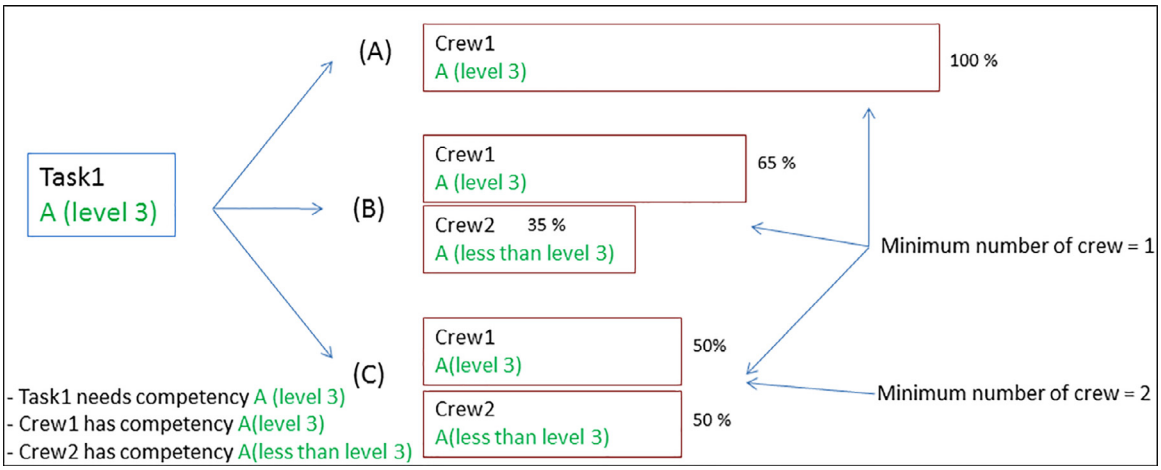

Fig. 2. Different possible scenarios for Crew competency.

The competence level should be maintained during the full duration of a task. This formulation ensures that at least one crew member has competence level 3 if multiple crew members work on the same task simultaneously:

$$\sum_n x_{nij} \cdot bm_{nk} \geq \frac{\sum_n x_{nij} \cdot bm2_{nk}}{d1_i} \quad \forall \quad i, j, k \tag{20}$$

2.5.5. Constraints in relation to transportation

These constraints ensure that a crew member is transported between the technical places that he works on during the day, and that he is transported to and from the depot at the start and the end of the shift. Each crew member works at the technical places that each allocated task belongs to:

$$w_{npj} \leq \sum_i z_{nij} \cdot tp_{ip} \quad \forall \quad n, p, j \tag{21}$$

$$w_{npj} \cdot M \geq \sum_i z_{nij} \cdot tp_{ip} \quad \forall \quad n, p, j \tag{22}$$

A crew member is only transported between the technical places that the tasks he is allocated are located:

$$\sum_q v_{npqj} \leq w_{npj} \cdot M \quad \forall \quad n, p, j \tag{23}$$

$$\sum_p v_{npqj} \leq w_{nqj} \cdot M \quad \forall \quad n, q, j \tag{24}$$

If a crew member works at more than one technical place during a shift, the technical places he is transported to and from while going between technical places are maintained by the following variables:

$$w1_{nqj} = \sum_p v_{npqj} \quad \forall \quad n, q, j \tag{25}$$

$$w2_{npj} = \sum_q v_{npqj} \quad \forall \quad n, p, j \tag{26}$$

Each crew member can only be transported to and from each technical place once per day:

$$w1_{npj} \leq 1 \quad \forall \quad n, p, j \tag{27}$$

$$w2_{npj} \leq 1 \quad \forall \quad n, p, j \tag{28}$$

If a crew member is working on a given date then he is transported only once from the depot and once to the depot:

$$\sum_p w_{npj} \cdot 2 - w1_{npj} - w2_{npj} = 2 \cdot y_{nj} \quad \forall \quad n, j \tag{29}$$

3. Proposed solution approach

The main goal of this work is to find feasible solutions for larger instances of the maintenance crew scheduling problem presented in the previous section, as the current practice is only able to solve problems with a planning horizon of two weeks. We propose a hybrid framework consisting of two phases, initial solution construction and a second phase of solution improvement. Previous work has shown that CP is an effective method for generating feasible solutions to highly constrained problems (Bockmayr & Hooker, 2005). Here we use Google's software suite for combinatorial optimisation (Google OR-Tools) Google (2012) to model the problem as a Constraint Satisfaction Problem (CSP). In the improvement phase, a MIP solver is used to further improve the initial feasible solution. Each phase is described in the following sections in more detail.

3.1. Construction phase

As mentioned above, we use CP to generate feasible solutions by modelling the problem as a CSP (Rossi, Van Beek, & Walsh, 2006). A CSP is a mathematical model described by three sets of elements: a set of variables, a set of possible values (domain) for each variable, and a set of constraints on the variables. Each solution is constructed by assigning values within the defined domain to the variables of the model such that every constraint is satisfied. The problem is modelled as a CSP with a customised global constraint added to deal with the specific crew competency constraints contained in the model. This process is illustrated in Fig. 3, inspired by Baptiste (2001).

As seen in Fig. 3, the process of solving a CP problem consists of four stages: problem definition, decision making, solution construction and defining the crew competency global constraint.

In the problem definition stage, in order to model the problem as a CSP, all of the MIP variables are defined over similar finite domains within a CSP model. All of the constraints except the constraints related to crew competency (18, 19 and 20 in Section 2.5.4 above) are defined as primary constraints. Due to the difficulty of satisfying the crew competency constraints, these are defined as customised global constraints in the final stage. Next in the decision making stage, we define the main decision variable and the way that the search tree is constructed. This is done by deciding on how we select the main decision variable and what value(s) are assigned to it at each node of the tree in order to branch the search tree. In the solution construction stage, at each node of the decision tree, one element of the main decision variable is selected and a value is assigned to it. Finally,

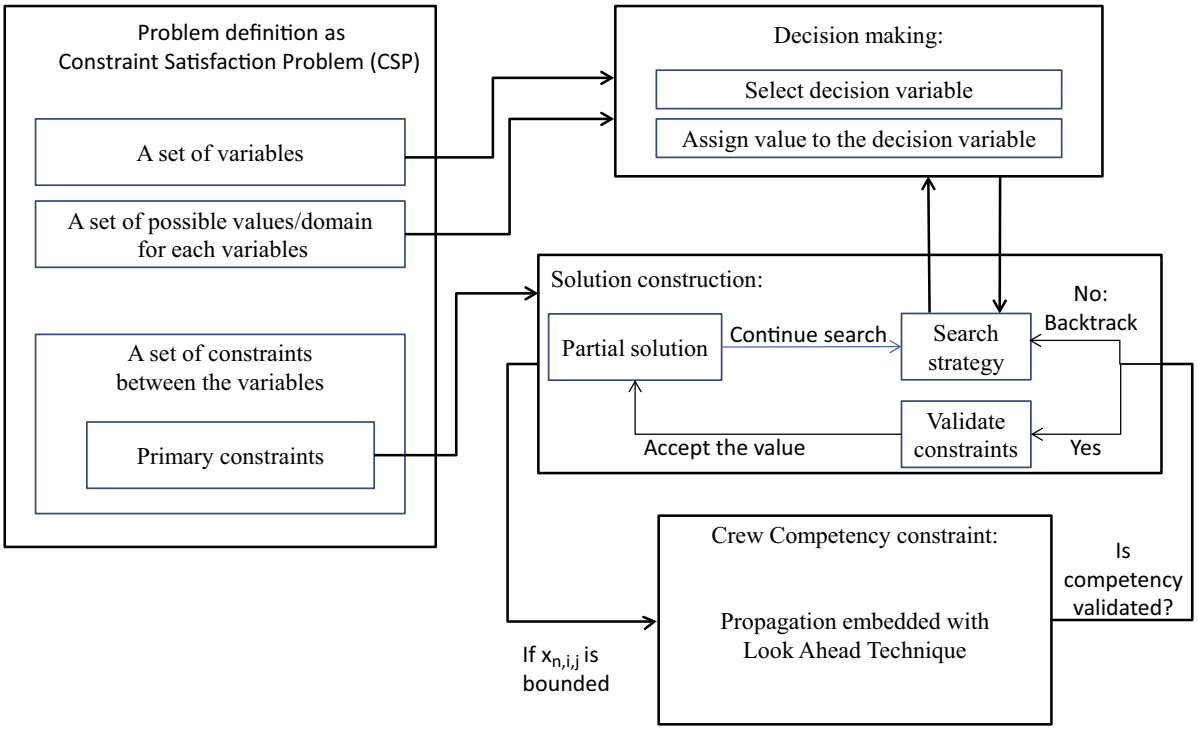

Fig. 3. Constraint programming framework.

by defining the crew competency constraints as global constraints, constraint propagation is used to make the given problem easier to solve. This is done by helping the solver to prune infeasible regions of the search space which violate the crew competency constraints. Infeasible areas are identified using a look-ahead technique embedded in a propagation algorithm.

The individual stages are described in detail in the following subsections.

3.1.1. Problem definition:

As this stage, all of the variables introduced in our mathematical model are defined as a set of variables in the CSP. The variables need to be scoped over finite domains. Consequently, the domain of each variable in our model is determined according to the domain of variables in the MIP model introduced in Section 2. The constraints can be defined as either initial/primary constraints or global constraints. Initial constraints can be defined as a set of $C = C_1, \ldots, C_K$ where each constraint comprises several variables and a list of values that the variables can take. From this perspective, the initial constraints correspond to what is known as a constraint in linear programming. In our model, all of the constraints except the constraints related to crew competency are defined as initial constraints.

A global constraint is defined as an "expressive and concise condition involving a non-fixed number of variables" according to the Global Constraint Catalogue (Beldiceanu, Carlsson, & Rampon, 2012). There are several well-known global constraints introduced in the literature which have been used in practice in many CP models (Aggoun & Beldiceanu, 1993; Beldiceanu, 2000; Caseau & Laburthe, 1997; Régin, 1994). In our approach, we have defined a customised global constraint composed of all of the related crew competency constraints in our mathematical model.

3.1.2. Decision making

The core decision variable of the problem is x_{nij}, which represents the fraction of task i fulfilled on date j by crew member n. Since most of the tasks are not atomic and need to be split over multiple days, the model mostly uses a fraction of the whole duration of each task. At each node of the tree, one variable from the x vector is selected and is given a value which propagates over the other variables in the search space. In Google OR-tools there are 16 strategies for selecting variables and 14 strategies for assigning values to a decision variable.

- Selecting decision variable: We have chosen the following five selection strategies, which all select the variable with the smallest domain: Min_Size, Min_Size_Lowest_Min, Min_Size_Highest_Min, Min_Size_Lowest_Max and Min_Size_Highest_Max. These five strategies only differ in the case of tie. Min_Size considers the order of variables in the vector, whilst the remaining four strategies select the variable with the lowest min value, the highest min value, the lowest max value and the highest max value, respectively.
- Assigning values to decision variables: After selecting a variable from x_{nij}, we should assign a value to it. We use two strategies strategies for assigning values: Min_Value and Max_Size. The former assigns the smallest possible value and the latter assigns the biggest value that is within the range of the selected variable in the vector.

We can see that the order of variables in x_{nij} has an effect on the strategies used to select the variable at each node in the case of a tie. According to the dimensionality of $x_{n,i,j}$, there are six possible orders that we can use: {i, j, n}, {i, n, j}, {j, n, i}, {j, i, n}, {n, j, i}, {n, i, j}. For instance, i, j, n denotes that the $x_{n,i,j}$ vector is generated by three inner loops with n being the most inner loop. In this way, we determine what portion of task i should be done by each crew member per day until the task is fully allocated i.e. the priority is on fulfilling tasks one by one per day by all crew members. As an example if n=3, i=2 and j=2, the vector of $x_{n,i,j}$ based on i, j, n order would be $x_{1,1,1}$, $x_{2,1,1}$, $x_{3,1,1}$, $x_{1,1,2}$, $x_{2,1,2}$, $x_{3,1,2}$, $x_{1,2,1}$, $x_{2,2,1}$, $x_{3,2,1}$, $x_{1,2,2}$, $x_{2,2,2}$, $x_{3,2,2}$.

With five selection strategies, six possible orders for the x vector, and two strategies for assigning values, we will test all 60 possible combinations of these three factors on a small problem

instance, to find the best combination before applying CP to larger problem instances.

3.1.3. Solution construction

In our framework, a systematic tree-based search strategy is used. At each node including the root, one variable from $x_{n,i,j}$ is selected and a value assigned to the chosen variable. In addition to the back-track technique embedded within CP, systematic search can be improved by look-back or look-ahead methods (Bayardo Jr & Schrag, 1997; Jussien, Debruyne, & Boizumault, 2000). In our framework, using the crew competency constraint as a customised global constraint helps the CP solver to prune infeasible regions of the search space violating this constraint. The infeasible areas are identified using a new look-ahead technique embedded in propagation algorithm explained below.

3.1.4. Crew competency global constraint

As mentioned previously, the most challenging part of this scheduling problem is satisfying all of the crew competency constraints. In CP, the solver treats a global constraint similarly to a primary constraint, in the sense that the class of global constraints is inherited from the same base class of primary constraints. When there is a change of variable domain or the bound of variable x_{nij}, an event is triggered which propagates its value on all other variables. The global constraint will register itself to this event and once the event is triggered the propagation algorithm associated with the proposed global constraint will be called.

Algorithm 1: Crew competency global constraint (part I - capturing the current state of the solution).

```
1  Initialise empty lists for boundedCrew, workingCrew,
   expertCrew, availableExperts
2  Initialise variables for total_crew_level, expert_duration,
   non_expert_duration, usable_expert_time
3  Other variables are as defined in the MIP model
4  if task i does not require any competencies then return
   success;
5  if task i is not compulsory then return success;
6  foreach crew ∈ N do
7      if (x_{crew,i,j} is bounded) then
8          add crew to boundedCrew
9          if (x_{crew,i,j} > 0) then
10             add crew to workingCrew
11             add crew competency level (bm3_{crew,k}) to
               total_crew_level
12         end
13         if (crew is expert) then
14             add crew to expertCrew
15             add x_{crew,i,j} to expert_duration
16         else
17             add x_{crew,i,j} to non_expert_duration
18         end
19     end
20 end
```

The overall process, presented in Algorithms 1 and 2, validates the crew competency constraints based on the current state of the solution and the potential future states that can be reached. The algorithm returns *fail* when either the crew competency constraints are violated, or it is deemed impossible to satisfy the crew competency constraints of task i, based on the availability of expert crew members (those who have at least competence level 3 for the competencies required for the task), when looking ahead at the possible future states of the solution. The algorithm returns *success* if the task is not compulsory (i.e. $x4_i$ is 0), if the task does not require any crew competencies or if it is possible to yield a feasible solution in future, with respect to the crew competency constraints, based on the expert crew members available.

Algorithm 2: Crew competency global constraint (part II - validating the crew competency with respect to the change in X_{nij}).

```
22 if all crew members are bounded then
23     if no crew member is working on task i then return
       success;
24     if total_crew_level < f then return fail;
25     if expertCrew list is empty then return fail;
26     if expert_duration < non_expert_duration / Σ_{n'} z_{n',i,j} then
       return fail;
27 else
28     if workingCrew is not empty then
29         max_additional_crew = d2_i − count(workingCrew);
30         if max_additional_crew == 0 then
31             if total_crew_level < f then return fail;
32             if expertCrew list is empty then return fail;
33             if expert_duration < non_expert_duration/Σ_{n'} z_{n',i,j}
               then return fail;
34             return success
35         end
36         foreach crew n' ∈ N, with competency k required for
           task i do
37             if n' is not in boundedCrew then
38                 if n' has unallocated time remaining on day j
                   then add n' to availableExperts;
39             end
40         end
41         if expertCrew and availableExperts are empty then
           return fail;
42         Sort availableExperts in ascending order of unallocated
           time remaining for t = 1 to
           Min(count(availableExperts), max_additional_crew) do
43             usable_expert_time += available time of t-th crew
               member in availableExperts list on day j;
44         end
45         potential_expert_duration = Min((c_i -
           non_expert_duration), usable_expert_time) +
           expert_duration;
46         if potential_expert_duration < non_expert_duration then
           return fail;
47     end
48 end
49 return success
```

As mentioned above, whenever x_{nij} is bounded or its domain is changed, the propagation algorithm will be called. It will first check if task i requires any competencies and whether or not it is compulsory to be completed (lines 4 and 5 in Algorithm 1). If not, it will return *success* and the solver can continue with the current state of x_{nij}. In both situations, as the solver does not need to validate crew competency constraints, these constraints are ignored.

When the algorithm does not return from either of the two situations above, it means that there is a need to validate the crew competency constraints when x_{nij} is changed. This is what the rest of the algorithm deals with, and is composed of the following two steps:

1. Capture the current state of the solution in terms of the resources required to validate the crew competency constraints (constraints 18, 19 and 20 in the MIP model). This part is presented in Algorithm 1 (lines 6-20).

2. Validate the crew competency constraints with respect to the change in x_{nij}. The pseudo-code of this part of the propagation algorithm is presented in Algorithm 2.

The current state of the solution is captured from lines 6 to 20. For each crew member, if the solver has decided whether crew member works on task i at date j or not (line 7), the crew member will be added to the *boundedCrew* list (line 8). If the crew member is working on the task (line 9), the crew member will also be added to the *workingCrew* list and their competency level ($bm3_{crew,k}$) is added to the *total_crew_level* variable (lines 10 and 11). Next, if the crew member is an expert in the competency required for the task (line 13), they will be added to the *expertCrew* list (line 14) and the time that the crew member spends on task i will be added to the *expert_duration* list (line 15). Otherwise the working time will be added to the *non_expert_duration* (line 17) as the crew member is not an expert in the competency required for this task.

Once the algorithm knows the current state of the solution being constructed, it can start validating the crew competency constraints with respect to the change in x_{nij}, as presented in Algorithm 2. At this point, there are two possible states that the solver can be in. Either the solver has already bounded all of the crew members for task i at date j (lines 22–26) or some crew members remain unbounded (lines 27–49).

If all crew members are bounded, the algorithm only needs to check the validity of the crew competency based on the current state as it is not possible to assign extra crew members to the task i on date j in future exploration of the search space. If no crew member is working on the task i (line 23), the algorithm will return *success*. Otherwise, it will check the crew competency constraints based on the current state of the solution, and will return *fail* in lines 24–26 if any of the constraints are violated (constraints 18, 19 and 20 from Section 2.5.4). If none of these constraints are violated, the algorithm will return *success* (line 50).

If the solver has not bounded all crew members for task i on date j, it means that it is possible at a future point in the search process to assign other crew members to complete the rest of the task. Consequently, a look-ahead technique can be used to monitor the feasibility of future assignments with respect to the crew competency constraints, by checking if the remaining expert crew members have enough free time to satisfy those constraints for this task. This allows us to prune infeasible areas of the search space in the case that the crew competency constraints cannot be met.

If there are any crew members working on the task i (line 28), the algorithm will calculate the maximum number of extra crew members who can be added to work on the task later (line 29). The number of additional possible crew members that can work on task i at date j, *max_additional_crew*, is calculated by subtracting the number of crew members who are currently working on the task from the maximum possible number of crew members that can work on the task together ($d2_i$). If this value is zero, it means that although there are crew members who are still unbounded, we have already assigned the maximum number of crew members for this particular task. In this case (line 30), the algorithm only needs to check the crew competency constraints (lines 31-33), without needing to look ahead to the future state of the solution. If none of these constraints are violated, the algorithm return *success* (line 34).

If it is possible to assign extra crew members to the task i on date j, the algorithm will use a look-ahead technique to consider the current and future state of the solution, based on the current value of x_{nij} in order to validate the crew competency constraints. The proposed technique guarantees that the feasibility of the solution is maintained from a crew competency point of view, following the change made to variable x_{nij}.

To provide the constraint solver with a better view of the availability of the other expert crew members to fulfil the rest of the task in future stages of the search, while satisfying the crew competency constraints, we first need to find the crew members who are expert in the competency required for task i who have free time available free time on date j (line 36 to 40). These crew members are added sequentially to a list of *availableExperts* (line 38).

If there are no crew members working on the task who are expert and no other crew members with the required expertise are available on date j, the algorithm will return *fail* as it is not possible to meet the crew competency constraints (line 41). This is effectively a look ahead technique for validating the crew competency constraints 18 and 19 in the MIP model. Otherwise, the algorithm sorts the list of *availableExperts* in ascending order of available time remaining on day j (line 42). Although we capture all of the free time of the expert crew members through *availableExperts* list, as there is a maximum number of crew members who can work on a task at one time ($d2_i$), we calculate the amount of expert time that can actually be added to the task (*usable_expert_time*). This is accumulated by looping over the minimum number between the count of *availableExperts*, and the number of crew members that can be added before exceeding the maximum crew capacity (*max_additional_crew*, calculated previously in line 29).

After calculating *usable_expert_time*, the algorithm checks how much of the task i can be undertaken by expert crew members in future, considering the actual time that task i requires to be completed (*potential_expert_duration*) (line 46). This is the minimum of the actual amount of the task which has been left undone by non-experts ($c[i]$ - *non_expert_duration*) and the free time of experts to undertake the task (*usable_expert_time*) added to the original amount of work undertaken on the task by experts (*expert_duration*). If the *potential_expert_duration* is less than the duration of non-experts (*non_expert_duration*), the algorithm returns *fail*. This is the last part of the look ahead technique which validates the final crew competency constraint 20 in the MIP model. If no constraint violations are identified by the previous validation checks, the algorithm will return *success* (line 50).

3.2. Improvement phase

Once a feasible solution has been found in the construction phase, a MIP solver starts searching in the branch and bound tree from that point and tries to improve the solution. Here we use CPLEX 12.4 to solve the MIP model as defined in Section 2. This process is known as a *warm start* (Gondzio, 1998). Feeding the MIP solver with a feasible starting solution helps the solver enormously by allowing for efficient cuts in the branch and bound tree, effectively reducing the size of the problem to such an extent that further search in the branch and bound tree becomes possible.

4. Results and discussion

In this section, we first introduce the four instances and then present the results of solving the problems by using the hybrid CP/MIP approach introduced above. We compare to both using a commercial MIP solver directly and modelling the problem as a Constraint Optimisation Problem (COP).

4.1. Dataset

The four instances used are based on real-world data provided by the Banedanmark planning department. In all four instances, there are the same 23 technical places and 8 crew members with

Table 1
Characteristics of the data instances used.

Instance Name	D2	D4	D6	D8
Horizon days	10	20	30	40
Working days	24	58	74	108
Number of tasks	11	39	47	59
Compulsory tasks	8	16	16	16
Tasks requiring competencies	10	34	41	53
Tasks $>$ 1 day long	6	15	20	26
Total duration (hours)	198.6	474.5	597.6	839.8
Minimum task duration (hours)	1.6	1.6	1.6	1.6
Maximum task duration (hours)	63.4	63.4	63.4	81.2

12 different crew competencies. Each task requires at most one competency. The closest task to the depot is 0.00 hours travel time (i.e. it is next to the depot), the furthest is 0.66 hours, and the average travel time is 0.28 hours from the depot. Table 1 presents the four different problem instances and their characteristics. The instances are named based on their planning time horizon, since they differ from one another with respect to the number of planning days (J), where each day is 6.90 hours long. The four problem instances, D2, D4, D6 and D8 have 2, 4, 6 and 8 week planning horizons, respectively. With eight crew members, each plan should have $J \times 8$ planning days in total, however, as not all crew members are available every day, the total number of available planning days for each instance is slightly less than this. There are different numbers of tasks in each instance, with the number of compulsory tasks to be scheduled in the plan, the number of tasks which last more than one working day and the number of tasks that require competencies also given. The total duration of tasks, and the minimum and maximum duration of a single task in each data instance are given in hours.

As seen in Table 1, the vast majority of tasks cannot be undertaken without an expert for a particular competency, adding to the complexity when scheduling crew members. Table 2 presents the number of tasks which require a specific competency and the number of crew members who have the required competency for each data instance. For instance, $D2$ includes tasks which require competency $A2$ (1 task), $B2$ (2 tasks), $B7$ (1 task), $B12$ (5 tasks) and $C11$ (1 task), with 5, 5, 4, 5 and 3 crew members having each of these competencies, respectively.

4.2. Tuning search in the decision making phase

In the decision making phase, we need to decide how to select the main decision variable and what value(s) are assigned to it at each node of the tree in order to branch the search tree. The first set of experiments investigates the performance of all possible combinations of the factors introduced in Section 3.1 on instance $D2$. Consequently, we can use the best tuning found to solve the larger problem instances. With five selection strategies, six possible orderings for the x vector, and two strategies for assigning values, we have tested all 60 possible combinations. Each combination is allowed to run for a maximum of 1 hour CPU time on a 2.1 gigahertz Intel Core i7-4600U CPU with 8.00 gigabytes RAM.

Assigning values using the Max_Size strategy does not generate any feasible solutions with any selection strategy and any ordering of the x vector within the time limit. This accounts for 30 of the 60 possible combinations tested. Considering the complexity of the model, the dependencies that exist, and the number of the variables we have, this is not a surprise since the Max_Size strategy leaves less room for assigning values to other variables. We also ran additional overnight experiments on a small number of combinations using the Max_Size strategy. However, in all cases no feasible solution was found for D2.

Moreover when using the Min_Size strategy, only three of the six orderings of the x vector are able to generate feasible solutions within the time limit: $\{i, j, n\}$, $\{i, n, j\}$, and $\{j, i, n\}$, ruling out another 15 of the combinations tested. We observe that these three orderings branch the search tree, prioritising finishing each task i over fully using the availability of each crew member n. As a feasible solution is found, more constraints have been propagated on the partial solution at each assignment by prioritising in this manner. This is likely to be due to the fact that there are more constraints on the tasks than the crew members. As x can propagate its value faster over a larger number of variables, the partial solution is constrained more quickly. Consequently, we are able to accept or refuse the partial solution at an earlier stage of the search.

This leaves 15 combinations of selection strategy, ordering and value assignment strategy which are able to produce feasible solutions. Table 3 shows the results of these combinations on instance D2, obtained using orderings $\{i, j, n\}$, $\{i, n, j\}$, and $\{j, i, n\}$ with five different selection strategies and Min_Size assignment strategy.

From this table, we can clearly see that the objective values obtained using different selection strategies are not significantly different from each other. Specifically, using $\{i, j, n\}$ and $\{j, i, n\}$ ordering, the objective values have the same values for all five selection strategies. For $\{i, n, j\}$ ordering, the objective values are 0.3714 for the *Min_Size*, *Min_Size_Highest_Min* and *Min_Size_Highest_Max* and 0.3655 for *Min_Size_Lowest_Max* and *Min_Size_Lowest_Max* strategies. Comparing the time taken to generate the first solution, $\{i, j, n\}$ is far quicker than the other two orderings, generating feasible solutions within 5 seconds for all five selection strategies. $\{j, i, n\}$ and $\{i, n, j\}$ take much longer to generate initial solutions, needing between 103 and 207 seconds and between 15 and 70 seconds, respectively. In addition, the number of failures (backtracks) and branches required to generate the feasible solutions for $\{j, i, n\}$ and $\{i, n, j\}$ is much larger than $\{i, j, n\}$. The large number of failures and branches indicates that when applied to larger instances, these two orderings may struggle to find a first feasible solution as they will not identify infeasible regions of the search space as quickly as $\{i, j, n\}$. As the primary goal of the constructive CP phase is to find a feasible solution, using a combination of strategies that minimise the time to find an initial solution is preferable. Hence we will use ordering $\{i, j, n\}$ with selection strategy Min_Size_Lowest_Min in the experiments on the larger instances in the next section.

4.3. Results and comparison

The hybrid framework we propose uses initial feasible solutions generated using CP as *warm start* solutions for an MIP solver. The MIP solver used is CPLEX 12.4 with default parameter settings. All experiments are performed on the same machine as above. We compare the quality of the solutions obtained by the hybrid CP/MIP framework to both solving the MIP model directly, and to improving the initial solutions obtained by CP by considering the problem as a Constraint Optimisation Problem (COP). Modelling the problem as a COP requires adding an extra constraint to find a solution with a better objective value than the previously found feasible solution (Rossi et al., 2006). For the hybrid CP/MIP and COP, the solvers are given 3 hours to improve the initial CP solution for each instance. In the case of the MIP solver only, it is allowed 3 hours CPU time.

Table 4 shows the objective function values and relative gaps of the solutions found by the CP/MIP hybrid, COP, and only the MIP solver for the four instances introduced in Section 4.1. In the results presented for the CP/MIP approach, the value of the initial feasible solution obtained by CP is given along with the value and relative gaps of the first, second and final solutions obtained by the MIP improvement phase. For COP the value of the improved solution after 3 hours is given, with the value obtained by feeding this instance to the MIP solver given in brackets for reference. Here we note that no optimisation is done by the MIP solver

Table 2
Competency-related attributes of the data instances.

Dataset		Competencies											
		A2	A3	B2	B4	B7	B9	B10	B12	C3	C4	C5	C11
D2	*Crew*	5		5		4			5				3
	Tasks	1		2		1			5				1
D4	*Crew*	5		5	5	4	5	5	5	5	5		3
	Tasks	3		4	1	1	1	3	8	6	3		4
D6	*Crew*	5		5	5	4	5	5	5	5	5	5	3
	Tasks	4		5	2	1	1	3	8	9	3	1	4
D8	*Crew*	5	5	5	5	4	5	5	5	5	5	5	3
	Tasks	7	1	6	2	1	1	3	8	15	3	2	4

Table 3
Results of feasible solutions found for instance D2, using three different orderings, five different selection strategies and Min_Size assignment strategy.

Selection variable strategy	Obj	Time_S	Failures	Branches
Order: i,j,n				
Min_Size	0.3753	2.71	95	304
Min_Size_Lowest_Max	0.3753	4.44	96	305
Min_Size_Lowest_Min	0.3753	1.98	96	305
Min_Size_Highest_Min	0.3753	2.20	95	304
Min_Size_Highest_Max	0.3753	3.25	95	304
Order: i,n,j				
Min_Size	0.3714	207.97	490515	981154
Min_Size_Lowest_Max	0.3655	142.09	496938	993999
Min_Size_Lowest_Min	0.3655	156.30	496938	993999
Min_Size_Highest_Min	0.3714	135.36	513396	1026916
Min_Size_Highest_Max	0.3714	103.45	513396	1026916
Order: j,i,n				
Min_Size	0.3711	29.12	114014	228142
Min_Size_Lowest_Max	0.3711	15.79	56820	113753
Min_Size_Lowest_Min	0.3711	70.05	56820	113753
Min_Size_Highest_Min	0.3711	29.08	114014	228142
Min_Size_Highest_Max	0.3711	22.61	114014	228142

for this result, the value is obtained by the pre-processing phase converting the COP result into a MIP model only.

A number of observations are worthy of mentioning here. On feeding the starting solutions provided by CP into the MIP solver, it can easily generate an initial feasible solution based on the CSP solution, improving that solution immediately. Additionally, in all four instances the relative gap to the lower bound is decreased considerably by the MIP solver. This is still true when the quality of the solution found is not improved, suggesting that the quality of the initial CSP solutions are good in these cases.

The only problem instance solved within the time limit using the MIP solver alone is the two-week problem (D2). It is interesting to note that in D2, where both the hybridised CP/MIP and MIP solver only methods end up with approximately the same result (0.3175 and 0.3173 respectively), the initial solution obtained by CSP is restricting the performance of the MIP solver in the hybrid CP/MIP approach to some extent.

For the 4, 6 and 8 week plans (D4, D6 and D8) the hybrid CP/MIP and COP approaches have feasible solutions generated in the construction phase. Comparing the quality of the best solutions obtained by COP and the CP/MIP hybrid, we see that the hybridised framework generates significantly better results, highlighted as bold in Table 4. In addition, the quality and the relative gap of the first solutions found by the cutting algorithms of the MIP solver, from both the CP and COP solutions, shows that using COP leads to limited improvement in objective value and relative gap compared to the original CP solution, despite the 3 hours computational time used by COP. For instance in D4, the objective value and the relative gap obtained on CSP and COP solutions are 0.3361 and 73.09%, and 0.3308 and 72.66%, respectively.

Table 5 reveals the computational time spent generating solutions for each of the three approaches tested. The computational time of the hybrid CP/MIP framework is the time spent generating the first feasible solution by CP added to the three hours time given to the MIP solver to optimise the solution. To evaluate how much time has been spent on the node relaxation and branching separately, we have distinguished between the time spent on each part in the table. Similarly, for the results using the MIP solver only, the time for both parts has also been included. For the COP solutions, the table shows the amount of time taken to generate the best solution within the time limit.

The time taken to generate the first feasible solution by CP is striking, where it takes approximately 2 seconds for D2 and 4.5, 12 and 52 minutes for D4, D6 and D8, respectively. It was not pos-

Table 4
Results of the hybrid CP/MIP framework, Only MIP solver, and COP (result fed to MIP) over all instances.

Instance	CSP + MIP		Only MIP		COP	
	Best integer	Rlt_Gap(%)	Best integer	Rlt_Gap(%)	Best integer	Rlt_Gap(%)
D2	0.3753(CSP)				**0.3674(COP)**	
	0.3688	60.67%	0.3571	17.90%	(0.3629	60.03%)
	2nd 0.3688	21.70%	0.3571	17.90%		
	Best 0.3175	**3.42%**	0.3173	3.89%		
D4	0.3663(CSP)		NA		**0.3610(COP)**	
	0.3361	73.09%			(0.3308	72.66%)
	0.3361	24.77%				
	Best 0.3162	**16.45%**				
D6	0.3392(CSP)		NA		**0.3389(COP)**	
	0.3166	74.89%			(0.3163	74.87%)
	0.3166	21.29%				
	Best 0.3138	**18.42%**				
D8	0.3290(CSP)		NA		**0.3270(COP)**	
	0.3130	79.31%			(0.3110	79.18%)
	0.3130	25.64%				
	Best 0.3130	**22.76%**				

Table 5
Time spent to generate solutions within the time limit by all three approaches: hybridised approach (CP/MIP), Only the MIP solver, and COP.

Instance	CSP + MIP			Only MIP	COP (within 3 hours)
D2	1.98 ≈ 2 seconds	Root_T:	2.57	3.87	284.908 ≈ 4.5 minutes
		B&C_T:	10579.8	10273.95	
		Total MIP:	10582.37 ≈ 3 hours	10277.81 ≈ 3 hours	
D4	256.318 ≈ 4.5 minutes	Root_T:	327.32		432.86 ≈ 7.2 minutes
		B&C_T:	10469.27		
		Total:	10796.6 ≈ 3 hours		
D6	724.776 ≈ 12 minutes	Root_T:	947.49		2599.574 ≈ 43.32 minutes
		B&C_T:	9850.2		
		Total MIP:	10797.69 ≈ 3 hours		
D8	3157.474 ≈ 52 minutes	Root_T:	8416.66		3524.647 ≈ 58.74 minutes
		B&C_T:	2380.89		
		Total MIP:	10797.55 ≈ 3 hours		

Table 6
Improvements made by COP to the original CP solution for each instance.

Instance	Obj	Time_S	Failures	Branches
D2	0.3753	1.98	96	305
	0.3741	7.82	32126	64367
	0.3713	27.79	165483	331084
	0.3674	284.91	1268374	2536868
D4	0.3663	256.32	110137	220992
	0.3646	258.80	110170	221059
	0.3636	261.31	110220	221159
	0.3631	263.85	110418	221558
	0.3615	266.66	110463	221650
	0.3612	269.60	111675	224075
	0.3611	425.62	500941	1002610
	0.3610	432.86	502184	1005093
D6	0.3392	724.78	724070	1449483
	0.3391	776.89	725395	1452134
	0.3389	2599.57	4662224	9325790
D8	0.3290	3157.47	372812	748162
	0.3280	3350.27	372857	748253
	0.3270	3524.65	373031	748602

sible for the MIP solver to find feasible solutions for data instances bigger than D2 at all. Note that, for the only data instance that MIP was able to generate solution (D2), we can see that feeding the MIP solver with the CSP solution leads to less root node processing compared to using the MIP solver alone. This indicates that starting with a feasible solution helps to reduce the time taken resolving the LP relaxation. Looking into the node processing time for all data sets, the increasing pattern is not a surprise when dealing with bigger data instances. Despite this reduction, continuous root relaxation still takes up a considerable proportion of running time in our model. For the D8 instance, it is worth highlighting that the node processing time has grown significantly. It is also notable that the MIP solver spends one fifth of its total execution time on the branching and cutting on such a big data instance. As this ratio is particularly high, it suggests that for this instance and any larger instances a longer running time might be more appropriate.

Looking at the time taken to find the best COP solutions for each data instance, we see that CP could not improve the CSP solution for the D2, D4 and D8 after a couple of minutes and for D6 after half an hour. This suggests that COP gets stuck in a local optimum quickly, long before reaching the time limit. Table 6 gives the details of the improvements made to the original CSP solution by COP during the 3 hour run for each instance. In this table, each row is representative of a feasible solution with the first solution corresponding to the original feasible CSP solution. Each subsequent row shows any improved solutions found by COP within the time limit.

Here we see that the first solutions (CSP solution) for all instances were yielded in 1.98, 256.32, 724.78 and 3157.47 seconds, respectively, for each instance. However, no solutions are improved further after 284.91, 432.86, 2599.57 and 3524.65 seconds by COP on D2, D4, D6 and D8, respectively showing that a large proportion of CPU time is spent without any improvement in quality observed. Comparing the number of failures and branches on the final solutions obtained by COP for D4 and D6 with those on earlier solutions we see that COP seems to get stuck in a local optimum. Moreover, comparing the quality of the first feasible solution with the quality of the best solution found over all instances shows a very small improvement has been made. Even though CP generates the first solution quickly, COP is not a good candidate approach to be used for the improvement phase.

Considering COP both quality-wise and time-wise, we found COP to be inferior to a commercial MIP solver when improving the initial solutions found by CP. Enhancing the initial solutions through COP demands more problem-specific customisation, consequently more implementation and development effort code-wise. For instance, employing local search instead of systematic search might improve the solutions, however this would require defining several neighbourhoods, due to the number of dimensions of the objective function. Additional effort would also be required for proper tuning within a framework such as a meta-heuristic or hyper-heuristic. The hybrid CP/MIP method takes advantage of the initial feasible solutions found by CP, eliminating large portions of the search space and resulting in smaller branch-and-cut trees. Passing the first found feasible solution as a starting solution to a MIP solver we are able to validate the quality of the initial solution and attempt to improve it using a MIP solver without having to tailor advanced, difficult to maintain heuristics to the problem.

5. Conclusion

In this paper, we have introduced a hybrid CP/MIP framework for solving a large scale maintenance crew scheduling problem for the Danish railway system. The model is based on a practical MIP formulation provided by Banedanmark, who are responsible for most of the railway infrastructure in Denmark. The problem involves a large number of real-life attributes and constraints, so the current practice of trying to solve the model directly using a standard MIP solver does not return any feasible solutions for planning horizons longer than two weeks. We have proposed a customised global constraint, embedded with a look-ahead technique in a CSP-based model, to construct initial solutions and attempt to improve them by warm-starting the MIP solver. The framework examines an exploration of variable/value ordering heuristics. Results have been presented using four real-world instances. The proposed hybrid CP/MIP framework has been shown to outperform both solving the problem as a MIP problem directly and using COP to improve the initial feasible solution found by CP.

The hybridised framework is a contribution to the development of integration between MIP and CP, where CP greatly reduces

the time required by the MIP to produce a solution. From a programming perspective, the framework is easy to maintain since the proposed propagation algorithm is logically and conceptually independent. This maintains the generality of the framework by focusing on feasibility checking, pruning infeasible areas from the perspective of crew competency constraints. If any other constraints need to be added to the model in future, it can be implemented as an independent constraint in the framework. Any new constraint simply needs to be added to the MIP model in the improvement phase.

In terms of future work, one limitation of the method proposed here is the transformation of a multi-objective problem to a single objective function. The weighted sum method used is based on expert opinion to reflect the importance of each component of the objective function. Future work will formulate this problem as a multi-objective problem directly, presenting and highlighting the different trade-offs that exist between multiple objectives. Our work here has also used a single MIP solver, under default parameter settings. As a wide range of commercial MIP solvers, with a large number of tunable parameters exist, another potential future research direction is the investigation of the ability of different solvers, using different parameter settings, to solve different instances of this problem.

References

Aggoun, A., & Beldiceanu, N. (1993). Extending chip in order to solve complex scheduling and placement problems. *Mathematical and Computer Modelling, 17*(7), 57–73.

Baldi, M. M., Heinicke, F., Simroth, A., & Tadei, R. (2016). New heuristics for the stochastic tactical railway maintenance problem. *Omega, 63*, 94–102.

Banedanmark., & Trafikministeriet. (2009). *The signalling programme : a total renewal of the Danish signalling infrastructure* p. 4. Banedanmark.

Baptiste, P. (2001). Combining operations research and constraint programming to solve real-life scheduling problems. [Online] www.ercim.eu/publication/Ercim_News/enw44/baptiste.html.

Bayardo Jr, R. J., & Schrag, R. (1997). Using csp look-back techniques to solve real–world sat instances. In *Proceedings of the AAAI/IAAI* (pp. 203–208).

Beldiceanu, N. (2000). Global constraints as graph properties on a structured network of elementary constraints of the same type. In *Proceedings of the international conference on principles and practice of constraint programming* (pp. 52–66). Springer.

Beldiceanu, N., Carlsson, M., & Rampon, J.-X. (2012). *Global constraint catalog, (revision a)*. Swedish Institute of Computer Science.

Bockmayr, A., & Hooker, J. N. (2005). Constraint programming. *Handbooks in Operations Research and Management Science, 12*(C), 559–600.

Bog, S., Nemani, A. K., & Ahuja, R. K. (2011). Iterative algorithms for the curfew planning problem. *Journal of the Operational Research Society, 62*(4), 593–607.

Borraz-Sánchez, C., & Klabjan, D. (2012). *Strategic gang scheduling for railroad maintenance*. Center for the Commercialization of Innovative Transportation Technology, Northwestern University.

Caseau, Y., & Laburthe, F. (1997). Solving small TSPS with constraints. In *Proceedings of the 14th international conference on logic programming* (pp. 316–330). MIT PRESS.

Cheung, B. S., Chow, K., Hui, L. C., & Yong, A. M. (1999). Railway track possession assignment using constraint satisfaction. *Engineering Applications of Artificial Intelligence, 12*(5), 599–611.

Gondzio, J. (1998). Warm start of the primal-dual method applied in the cutting–plane scheme. *Mathematical Programming, 83*(1–3), 125–143.

Google (2012). Google Optimization Tools. [Online] http://developers.google.com/optimization/.

Gorman, M. F., & Kanet, J. J. (2010). Formulation and solution approaches to the rail maintenance production gang scheduling problem. *Journal of Transportation Engineering, 136*(8), 701–708.

Jussien, N., Debruyne, R., & Boizumault, P. (2000). Maintaining arc-consistency within dynamic backtracking. In *Proceedings of the international conference on principles and practice of constraint programming* (pp. 249–261). Springer.

Khalouli, S., Benmansour, R., & Hanafi, S. (2016). An ant colony algorithm based on opportunities for scheduling the preventive railway maintenance. In *Proceedings of the 2016 international conference on control, decision and information technologies (CODIT)* (pp. 594–599). IEEE.

Lidén, T. (2015). Railway infrastructure maintenance-a survey of planning problems and conducted research. *Transportation Research Procedia, 10*, 574–583.

Nemani, A. K., Bog, S., & Ahuja, R. K. (2010). Solving the curfew planning problem. *Transportation Science, 44*(4), 506–523.

Peng, F., Kang, S., Li, X., Ouyang, Y., Somani, K., & Acharya, D. (2011). A heuristic approach to the railroad track maintenance scheduling problem. *Computer-Aided Civil and Infrastructure Engineering, 26*(2), 129–145.

Peng, F., & Ouyang, Y. (2014). Optimal clustering of railroad track maintenance jobs. *Computer-Aided Civil and Infrastructure Engineering, 29*(4), 235–247.

Régin, J.-C. (1994). A filtering algorithm for constraints of difference in CSPS. In *Proceedings of the AAAI: 94* (pp. 362–367).

Rossi, F., Van Beek, P., & Walsh, T. (2006). *Handbook of constraint programming*. Elsevier.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., & Dueck, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics, 159*(2), 139–171.

Wen, M., Li, R., & Salling, K. B. (2016). Optimization of preventive condition-based tamping for railway tracks. *European Journal of Operational Research, 252*(2), 455–465.